Using variational autoencoders(VAEs) for data preprocessing

Andrew Zhuravchak

***Abstract*—Data preprocessing is an essential part of data analysis. In this study, I propose the usage of Variational Autoencoders(VAEs) for data augmentation and noise reduction problems on the MNIST dataset. Further, a similar approach can be used for other data preprocessing techniques.**

I. Introduction

Variational Autoencoders (VAEs) are powerful generative models, that have applications as diverse as from generating fake human faces, to producing purely synthetic music. Also, they work better than any other model when you'd like to alter or explore variations on data you already have, and not just in a random way either, but in a desired, specific direction.

An autoencoder network is a pair of two connected networks, an encoder and a decoder. An encoder network takes in an input and converts it into a smaller, dense representation(latent space), which the decoder network can use to convert it back to the original input.

The fundamental problem with standard autoencoders, for the generation, is that the latent space they convert their inputs to and where their encoded vectors lie, may not be continuous, or allow easy interpolation.

Variational Autoencoders (VAEs) have one fundamentally unique property that separates them from vanilla autoencoders, and it is this property that makes them so useful for generative modelling: their latent spaces are, by design, continuous, allowing easy random sampling and interpolation.

We can achieve this by encoding not just in vector(of size n) of plain numbers, but in distributions, which we describe by two vectors of size n: a vector of means, μ, and another vector of standard deviations, σ. This means, that even for the same input, the actual encoding can differ a little bit(depending on standard deviation) and it adds a stochastic part to the model.

II. Used VAE model and its architecture

The whole VAE architecture, used in this work, is pretty simple and consists of 2 layers in decoder/encoder and 2 layers for variances and means, which describes the distribution and takes a role of the latent space. In both cases(data augmentation and noise reduction) was used the same VAE with the same architecture.

The loss function, which was used in VAE consists of BCE and KLD terms:

- BCE(Binary Cross Entropy) - trying to make our reconstruction as accurate as possible

$$\sum_{y=1}^{n} (y_i \ln p_i + (1 - y_i) \ln 1 - p_i$$

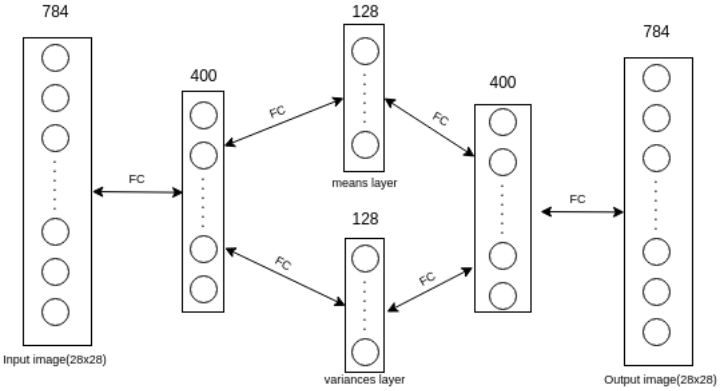

Fig. 1. VAE architecture

- KLD(KullbackLeibler divergence) - describes how much does one learned distribution deviate from another, in this specific case the learned distribution from the unit Gaussian. So is tries to push the distributions as close as possible to unit Gaussian

$$\frac{1}{2} \sum_{j=1}^{n} (\mu_j^2 + \sigma_j^2)$$

To implement encoder and decoder as a neural network, we need to backpropagate through random sampling and that is the problem because backpropagation cannot flow through a random node. To overcome this obstacle, we use reparameterization trick. It consists of saying that sampling from $z \sim N(\mu, \sigma)$ is equivalent to sampling $E \sim N(0, 1)$ and setting $z = \mu + \sigma \odot E$. As epsilon can be seen as an input of the network, you sampled while keeping your sampling operation differentiable.

III. Data Preprocessing with VAEs

A. Data augmentation

When we need to build an accurate model, it happens that there is a lack of data and model fails on test data. In a such case, we are trying to enlarge a dataset to achieve better model generalisation. Along with common techniques, we can apply VAEs for this data augmentation problem.

The main idea behind is that on training step, the model will learn some key attributes of the data. So, later during the generation mode by adding some noise we could generate pretty similar data.

I applied this technique for MNIST dataset and enlarged the training dataset two times. I have used a pretty simple CNN model for image classification. Accuracy on test data:

- Original dataset: 98.9%
- Generated dataset: 98.4%
- Both datasets together: 99.1%

You can see the example of this model usage on Fig.2 (original images at the top and generated at the bottom)

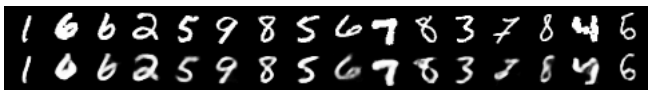

Fig. 2. Data augmentation

B. Noise reduction

While preparing the dataset with images for real usage it's a common problem to remove the noise from them. I proposed the use of VAE in this case. For images from MNIST dataset we randomly add gaussian and speckle noise. During the training, it takes two versions of images(the noisy and original one) and tries to map them with each other.

After that on test mode, we have a pretty good working noise reduction model. On the Fig.3, you can see the usage of this model.

- Left side: original image
- Middle: "noisy" image
- Right side: "denoised" image

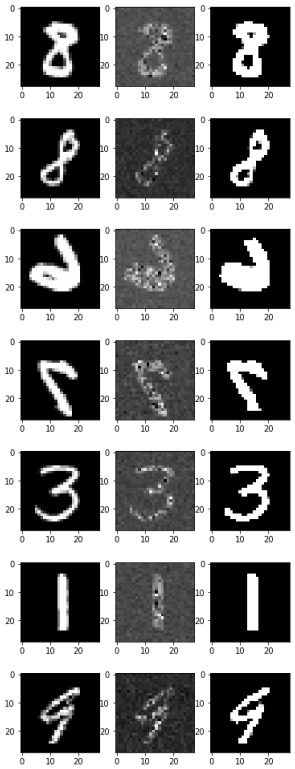

Fig. 3. Noise reduction

IV. Results and conclusions

The main idea behind this work was to demonstrate the real usage of VAEs in data preprocessing and their potential implementations in further investigation. I have shown two examples with data augmentation and noise reduction and VAE successfully dealt with both of them. Probably, in further studies, it'd better to take not just some common dataset(like a MNIST), but also something very specific with usage in real-life. In such case, we would see more practical VAE application.

References

[1] Irhum Shafkat. Intuitively Understanding Variational Autoencoders. https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

[2] What is a variational autoencoder? https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

[3] Javier Jorge, Jesus Vieco, Roberto Paredes, Joan Andreu Sanchez and Jose Miguel Bened. Empirical Evaluation of Variational Autoencoders for Data Augmentation. http://www.scitepress.org/Papers/2018/66186/66186.pdf

[4] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, Ian Reid. A Bayesian Data Augmentation Approach for Learning Deep Models. https://arxiv.org/pdf/1710.10564.pdf1